CMPT 280

Topic 6: Trees

Mark G. Eramian

University of Saskatchewan

References

- Textbook, Chapter 6

Reading Refresher

- In general, how many children may a tree node have?
- What is a binary tree?
- Define ancestor, descendent, sibling, level, and height.
- What is a leaf node? Internal node?
- Why is a tree considered a container?
- What pieces of data must be in a binary tree node?

Implementing a Simple Tree

- Our goal for this topic is to finish implementing the simple linked tree that was begun in Chapter 6 of the textbook readings.

A Simple Binary Tree ADT

Name: SimpleTree<G>

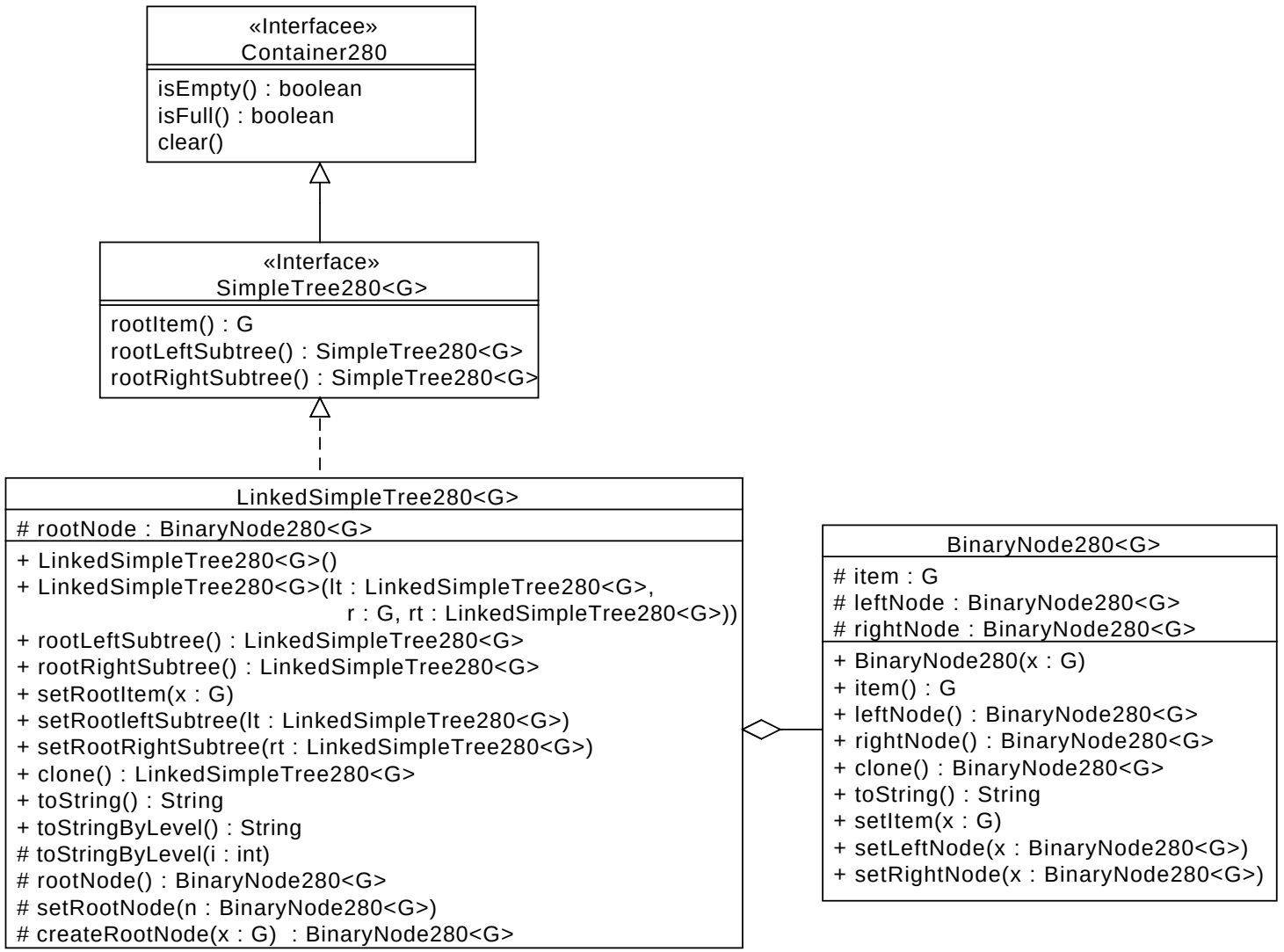
Sets:
T : set of trees containing elements from G
G : set of elements allowed in the trees
B : $\{\mathbf{true}, \mathbf{false}\}$

Signatures:
newTree<G> : $\rightarrow T$
T.initialize(t_1, g, t_2): $T \times G \times T \rightarrow T$
T.isEmpty: $\rightarrow B$
T.rootItem: $\not\rightarrow G$
T.rootLeftSubtree: $\not\rightarrow T$
T.rootRightSubtree: $\not\rightarrow T$

Preconditions: For all $t \in T$
t.rootItem: t is not empty
t.rootLeftSubtree: t is not empty
t.rootRightSubtree: t is not empty

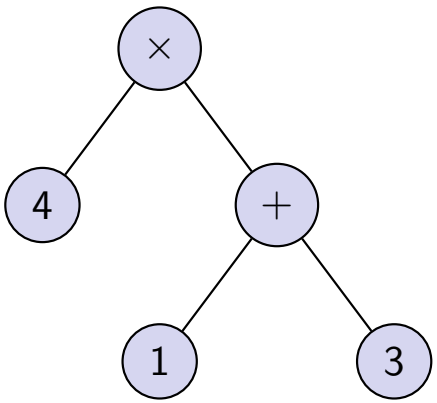
Semantics: For $t, t_1, t_2 \in T$, $g \in G$
newTree<G>: construct a new empty tree to hold elements from G.
t.initialize(t_1, g, t_2): initialize t to have root g, left subtree t_1 and right subtree t_2
t.isEmpty: return true if t is empty, false otherwise
t.rootItem: returns the root element of t.
t.rootLeftSubtree: returns the left subtree of t.
t.rootRightSubtree: returns the right subtree of t.

Specification of a Linked Tree

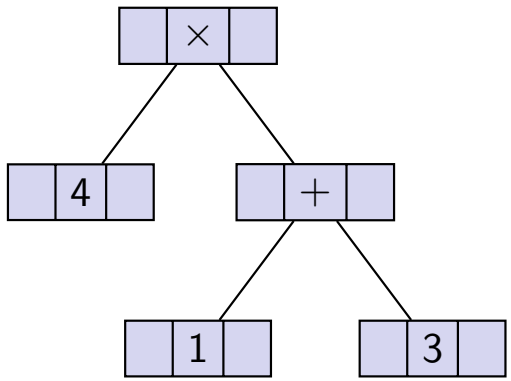


Views of Linked Representation of a Binary Tree

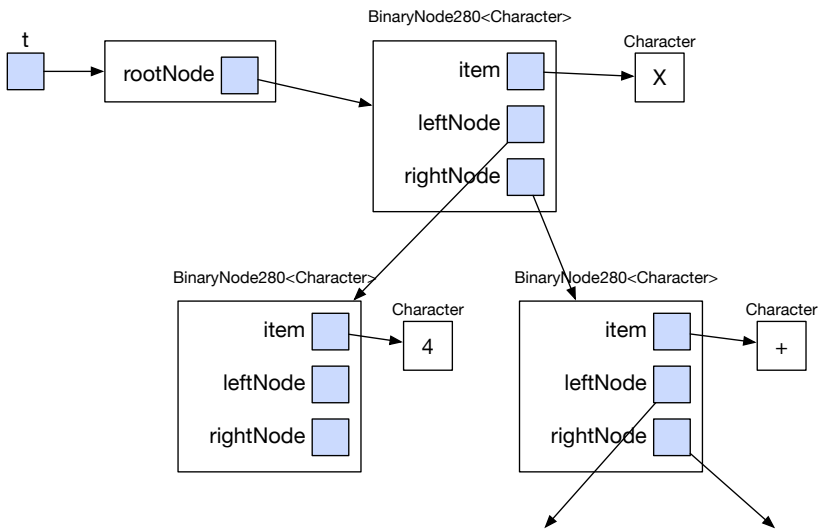
Conceptual View:



Structural View:



Implementation View:



Exercise 1

- Write the class header for `BinaryNode<I>`
- Write the declaration of the instance variables for `BinaryNode<I>`.
- Write the headers for the constructor and methods of `BinaryNode<I>`.

(In other words, write everything for the class except the method bodies.)

Demo 1

- Let's write the `item`, `leftNode`, and `rightNode` methods...

Exercise 2

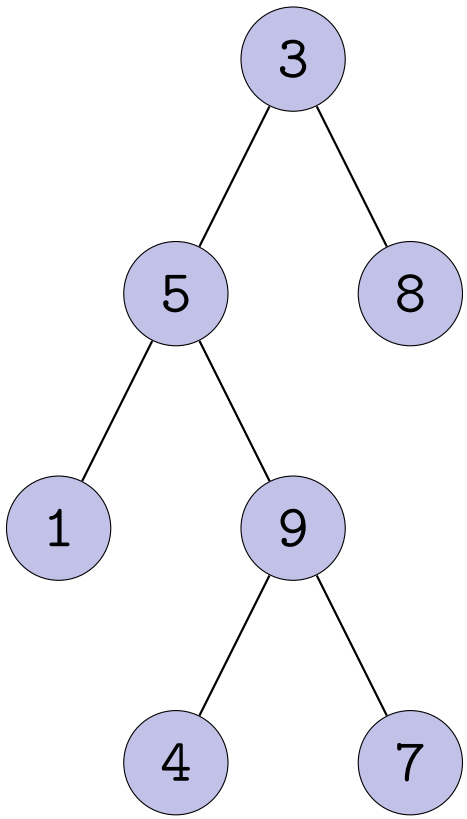
- Fill in the method bodies for `setItem`, `setLeftNode`, `setRightNode`, and `toString`.

Exercise 3

- Implement the `LinkedSimpleTree280<I>` class.
- We'll start off working together, then we'll break into small groups for implementing some of the methods.

Exercise 4

- Write program that will build the following tree using `LinkedSimpleTree280<I>`:



Next Class

- Next class reading: Chapter 7: Cloning **and** Chapter 8: Tree Traversals.